COMPUTER SYSTEMS LABORATORY

STANFORD UNIVERSITY · STANFORD, CA 94305-4055

Microsupercomputers: Design and Implementation

Stanford University
Computer Systems Laboratory

Technical Progress Report

November 1988 - March 1989

Principal Investigator
John L. Hennessy

Associate Investigator
Mark A. Horowitz

Technical Progress Report

November 1988 - March 1989

Contract No. N00014-87-K-0828

Order No. 1133

R & T Project Code: 4331685

Principal Investigator: John Hennessy

Monitored by M. Pullen, J. Toole

Dist. "A" per telecon Dr. Andre van Tilborg. Office of Naval Research/ code 1133.
VHG 11/13/90

Accesion For	
NTIS CRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By per call	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

This work is supported by the Defense Advanced Research Projects Agency and Office of Naval Research.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Technical Progress

This report summarizes our progress for the period November 1988 - March 1989.

1 Parallel Processor Architecture

1.1 Basic Architecture Studies

In the parallel architecture area, one of our main focuses has been on obtaining multiprocessor memory reference and synchronization traces [Davis 88] from several *real* parallel applications, and using them to evaluate implications for parallel architectures. Our original studies were done using traces obtained from a VAX-8350 (using modified microcode). A major limitation of the microcode-based scheme was that it was not possible to get traces for more than 4 processors. Subsequently, we developed and used a scheme based on the VAX T-bit. This enabled us to use an arbitrary number of processes but it was too slow. It took approximately a week to get a reasonable length 32 process memory reference trace. We have developed a new and powerful tool called Tango to help with this task.

Tango is a software tracing and simulation system that provides data to aid in evaluating parallel programs and multiprocessor systems. The system provides a simulated multiprocessor environment by multiplexing application processes onto a single processor. Tango allows the user to trace the shared memory and synchronization behavior of parallel programs. The system is efficient, and can be used with a wide range of machine and programming models. Accurate tracing is difficult since parallel programs are typically non-deterministic: the execution path through the program depends on the real time behavior of the hardware system. Tango offers accurate tracing by allowing the user to optionally integrate his shared memory and synchronization timing simulations into the tracing environment.

The Tango system gains efficiency by focusing on the application behavior specifically related to our multiprocessor behavior, and also by running compiled code, rather than emulated code. In using Tango, the application source code is not modified; it is automatically augmented in the compilation process to produce a compiled simulation. The compiled code includes the tracing and simulation code needed for a particular set of studies-and is run directly on an available uniprocessor-there is no time expensive instruction emulation, and the simulation system is tailored to the events of interest. The Tango system currently runs on the MIPS M120 and on the DECStation 3100. It is about 100-1000 times faster than our old T-bit based trace generator, and thus qualitatively changes the kinds of studies that we can do. We have used this version to trace the shared memory reference behavior of several of our parallel applications with up to 100 processors.

1.2 Scalable Shared Memory Multiprocessors

As stated in our previous report, we are actively pursuing the design of a large scale shared-memory machine using the fastest available microprocessors. The main technique that we are exploiting to achieve scalability is that of directory-based cache coherence. The basic principle is that each memory line keeps track of processors that are caching it. Consequently, point to point invalidation messages are sent only to those processors that are actually caching the data.

This is in contrast to snoopy schemes where invalidation/update broadcasts are used to keep caches consistent, thus consuming large amounts of bandwidth.

Several decisions have become more concrete since our last report and several new issues have cropped up. We have decided to use the Silicon Graphics Iris-4D workstation as the individual node of our multiprocessor. The Iris workstation is itself a multiprocessor consisting of 4 processors, with each processor having a 25MHz MIPS-R3000 as an integer unit and a MIPS-R3010 floating point unit and a large 2-level cache. Thus, the individual node in our machine will have about 80 MIPS of integer performance and about 16-20 MFLOPS of floating point performance. The main reason for going with a commercial machine as the node of our multiprocessor was to reduce the amount of hardware we needed to build at Stanford, thereby reducing the time to completion of the machine. It was a complex trade-off, since we do lose some design flexibility, but we felt it was worth it.

In the past six months we have been working on finalizing the directory-based cache coherence protocol. We have moved away from the blocking protocol we had originally proposed to a non-blocking protocol. In the non-blocking protocol, when a request for a location that is dirty in a remote cache is received by a directory, the directory forwards that request to the remote cluster rather than holding on to the request. We believe this protocol will significantly reduce the buffering requirements and enhance the performance of the machine. We are currently doing simulations to confirm this intuition.

We have also had to make decisions regarding the degree of shared-memory consistency that our machine will have. We are moving away from the traditional notions of strong consistency and weak consistency to a still weaker form of consistency. The key idea is that you have to wait for the writes to complete only before a sychronization instruction that can release another processor, and not before all synchronization instructions. We expect our protocol to offer higher performance and simpler implementation. We have also been working on providing hardware support for high contention synchronization operations like barriers. Detailed design of all critical parts of the machine, with the SGI Iris-4D as the base, is now proceeding.

We are also continuing our studies about invalidation patterns in multiprocessors. Since our last report, we have gathered additional information about the affect of varying the cache line size on invalidation patterns. Our studies show that both line sizes that are too small and line sizes that are too large can cause a higher number of invalidations. In addition, large line sizes can cause a greater proportion of invalidations that must be sent to many processes, that is, there is a greater proportion of references that invalidate a large number of caches. Compiler support can aid in the selection and placement of data objects with respect to cache lines, and we are exploring such possibilities.

2 Parallel Software

One of our main goals in the parallel software area is to make sure that when the parallel multiprocessor we are building comes up, there are significant applications and software that we can run on it. To this end, we have been studying several compute-intensive applications and working on parallelizing them. One of the new applications we have decided to parallelize is SUPREM-IV, which is an integrated circuit fabrication simulator developed at Stanford by Mark Law and Connor Rafferty. It is capable of modeling etching, depositing, diffusion, oxidation, and implants. The application is very time consuming and can easily take several hours of time on powerful workstations. The underlying model requires solving a set of equations on a two-dimensional grid using the preconditioned conjugate-gradient method. We have an initial implementation of the core of SUPREM-IV now running on the Encore Multimax. We are working on refining this implementation further, evaluating the scalability of the parallel implementation, and studying how locality in the computation can be increased so that it works well on our directory-based multiprocessor. We will be using TANGO and the architectural simulator discussed in Section 1.1 to carry out this work. We are also looking into general sparse matrix packages and studying how they can be implemented well on our directory-based multiprocessor, or alternatively, what changes do we have to make to our architecture so this large and important class of applications can be run well.

Another application area that we have been exploring is digital logic simulation. We are exploring the use of the Chandy-Misra distributed simulation algorithm as applied to the domain of logic simulation. We have now gathered up several realistic benchmark circuits, and have a parallel implementation running. In [Soule 89a], we present data characterizing the intrinsic parallelism in the benchmark circuits using the generic Chandy-Misra algorithm. Our results show that the average number of logic elements available for concurrent execution ranges from 6.2 to 92 for the benchmark circuits, with an overall average of 50. Although this is twice as much parallelism as that obtained by traditional event-driven algorithms, we feel it is still too low. One major factor limiting concurrency is the large number of global synchronization points --- "deadlocks" in the Chandy-Misra terminology --- that occur during execution. Towards the goal of reducing the number of deadlocks, we present a classification of the types of deadlocks that occur during digital logic simulation and some domain specific methods for reducing them. Since the last report, we have implemented most of the proposed methods for reducing number of deadlocks. The results are quite promising and are presented in [Soule 89b].

One of our new efforts in parallel software has been in the area of process control and scheduling issues for shared-memory multiprocessors. The target environment we consider is a shared-memory multiprocessor on which muiltiple users are developing and running their parallel applications at the same time. In such an environment, where the machine load is continuously varying, how should an application maximize its performance while being fair to other users of the system? Our preliminary results are presented in [Tucker 89]. We first show that if the number of processes belonging to a parallel application significantly exceeds the effective number of physical processors it is getting, then its performance can be significantly degraded. The degradation may be due to preemption of processes while they are within critical sections, due to bad performance of barriers under overload, and also due to the overheads of unnecessary context switching and processor cache corruption. We propose a way of controlling the active number of processes associated with an application dynamically to ensure good performance. A preliminary implementation of the proposed scheme is now running on the Encore Multimax and we show how it helps improve the performance of several applications. In some cases, the improvement is more than a factor of two.

3 Uniprocessor Architecture

3.1 Super Scalars

We have continued our work on investigating how much parallelism is available at the lowest level -- in the base instruction stream of a processor. Our initial work in this area [Smith 89] showed, for the non-scientific applications we are interested in, it is possible to execute a program in about one-half the number of cycles needed by a conventional RISC machine. This work pointed out that the main problem was in the area of instruction fetching and not really contention for functional units.

We have continued our work investigating both hardware/software tradeoffs in this area. Using a trace-driven simulation system we have explored the performance of different hardware fetch models as well as different methods of building the execution units. This data has provided a much better feel for what really matters in this class of processor. The results indicate there are four main features needed to achieve a speed-up of two. They are branch prediction, a four wide instruction decoder, out-of-order execution, and register renaming. The first two are needed to get enough instructions into the execution unit and the latter two are needed to remove extraneous dependencies to allow greater parallelism.

The out-of-order issue allows an instruction *B* that logically follows *A* to be issued before *A* if it does not have any data dependencies. Register renaming increases the available parallelism by removing data dependencies that arise because of storage conflicts, caused by register reuse. To implement this rearrangement of instructions, the hardware must solve two problems. First it must be able to detect and track data dependencies, since this information now sets the actual instruction sequence. Second, it must store the results from the instructions in such a way that the true sequential state of the machine can be reconstructed in the case of an exceptional condition occurs (like an interrupt). Neither of these tasks are simple. We are currently looking at the hardware cost of these features, and methods of simplifying their implementation.

The complexity of the hardware needed to implement out-of-order issue has increased our interest in a hardware/software solution, where some of the complexity of the hardware can be migrated into the compiler/reorganizer system. This work is just beginning and will use the previous work as the baseline for comparison.

3.2 High-Performance Cache Design

Traditionally, caches are evaluated on the basis of time-independent metrics, such as miss rates and traffic ratios. However, computers as a whole are compared on the basis of overall performance. When the basis of comparison of caches is changed miss ratios to execution time, a whole new set of tradeoffs between the traditional organizational parameters - cache size, associativity and block size - and the temporal parameters that the system designer has at his or her disposal, in particular the cycle time, is exposed.

The primary consequences for cache design from this shift in perspective are: i) a clear optimum cache size exists for each implementation environment (for most situations, that optimum lies between 32KB and 128KB); ii) for cache sizes over 16KB, set associativity improves performance only if the implementation of associativity degrades the cycle time by less than 4ns over that of a comparably sized direct-mapped cache; iii) the performance-optimal block size is primarily dependent on the main memory characteristics, and for most systems that is either four or eight words; and iv) most complicated fetch strategies do not improve performance very much because they either don't have enough time to fetch something before it's needed, and when there is time to fetch something useful, it is unclear what should be fetched. Fundamentally, this is because cache misses are highly clustered in time. These results are presented in more detail in [Przybylski 88a, Przybylski 88b].

The perspective of performance-directed cache design was also extended into the realm of multi-level cache hierarchies. The presence of a second-level cache can decrease the optimum size and cycle time of the first-level cache, and significantly improve performance beyond the best attainable with a single level of caching. The optimal characteristics of the second level-cache depend on the miss ratio of the first-level cache, but in general an optimal second level cache will be significantly larger and more likely to be assocaitive than if it were alone in the system. This is because the presence of the first-level cache reduces the number of accesses to the second level without significantly reducing the number of misses in that cache. This shifts the tradeoff away from short cycle times and towards low miss ratios [Przybylski 89]. Given a selection of implementable caches of all sizes, dynamic programming can be used to select the overall best memory hierarchy.

4 Computer-Aided Design (CAD) Tools

4.1 Computer-Aided Synthesis

This work yielded a major breakthrough in optimal logic synthesis of digital synchronous sequential circuits [De Micheli 89]. We have developed algorithms for minimizing i) the area of synchronous combinational and/or sequential circuits under cycle time constraints and ii) the cycle time under area constraints. Previous approaches attacked this problem by separating the combinational logic from the registers and by applying circuit transformations to the combinational component only. We have shown instead how to optimize concurrently the circuit equations and the register position. This method is novel and achieves results that are at least as good as those obtained by previous methods. A computer implementation of the algorithms in program MINERVA has been accompolished and experimental results have supported the theory.

4.2 Automatic Layout

The aim of the Locus project is to obtain high quality automatic placement of integrated circuits by combining information from the routing into the placement optimization process. Because routing is a time-consuming task, the first phase of this work was to produce a parallel global router (for standard cell design technologies) that can perform the routing very quickly. The parallel router, LocusRoute, is complete, having obtained significant speedups (10 to 13 times faster) on a 16 processor Encore MULTIMAX. We anticipate that it will achieve significant speedup on more than 100 processors. This work has been published in [Rose 88a], which focuses on the CAD algorithm, and in [Rose 88b] which discusses the parallel aspects. Improvements to the basic algorithm and a more sophisticated multiprocessor scheduler have recently been added. A revised publication that contains all aspects of the router and the new improvements has been submitted [Rose 89a].

Work continues on the second phase, the Locus placement program. The input parsing, and initial cost function calculation (including an entire routing of the starting placement) are complete. An optimization strategy based on top-down N-way partitioning integrated with complete routing is now being devised. This involves the design of the cost function, the move sets, the partitioning line sequence and the basic optimization process, which will be based on Simulated Annealing. The temperature schedule will make use of the temperature measurement scheme discussed below.

4.3 Temperature Measurement of Simulated Annealing Placements

One way to alleviate the high computational cost of Simulated Annealing is to replace part of it with a faster heuristic, and then follow this with lower-temperature Simulated Annealing. A crucial parameter in this kind of approach is the temperature at which to begin the Simulated Annealing phase. This work addresses that problem, in the context of the automatic placement, by developing a method of measuring the temperature of a given placement.

Recently we have formulated a more precise mathematical representation of the temperature measurement method, based on the Markov Chain transition probabilities of the Simulated Annealing process. We have learned that, while the method is theoretically fallible and may result in misleading temperatures, in practice for the placement problem it gives good answers. An early version of this work was published in [Rose 88c]. The newer work is submitted for publication in [Rose 89b].

We have recently learned that two others efforts are in the process of applying this research - in research at Yale University and in commercial development at the CADENCE company in San Jose, CA.

4.4 Ariel

Ariel, our system for analyzing voltage drops and current density in the power buses of VLSI circuits, has been modified to solve networks of arbitrary topology [Stark 89]. It accomplishes this by separating out simple subnetworks whose current-voltage characteristics can be calculated independently of the rest of the network, then solving the remainder using standard sparse positive definite network analysis.

Ariel has also been extended to analyze ECL circuits. A static checker that analyzes an ECL network, calculates the voltage ranges over which all nodes in the circuit can swing, and determines where currents enter and leave the power network has been written. This current-distribution information is fed into Ariel, which calculates the corresponding voltage drops and current densities for the power buses.

5 VLSI

We have been continuing our efforts to exploit the capabilities of integrated circuit technologies for high speed systems. During this period we have continued our efforts in BiCMOS, testing and programmable logic.

5.1 BiCMOS

Our work in BiCMOS has continued in three fronts: fast sRAMs, innovative BiCMOS logic circuits, and CAD tools for BiCMOS. In the area of fast sRAM we were very pleased to find out that our new BiCMOS memory cell design, the CSEA cell, was used by a commercial company, Aspen, in their 3ns 4K BiCMOS RAM [Cole 89]. Our work on a 64k BiCMOS RAM is proceeding. We are working with Texas Instruments' 0.8μ BiCMOS technology and hope to have a chip ready for fabrication by the end of the academic year.

The major work on BiCMOS circuits this past period was to design a test mask for the Stanford BiCMOS process. This technology is interesting since it uses triple-diffused bipolar transistors, rather than adding an epi layer. Not using epi is quite controversial, but we think the advantages of smaller, low capacitance devices might overcome the disadvantages of the large collector resistances. The devices seem especially promising as the minimal feature sizes approach 1μ. The test mask contains a number of innovative circuits including sRAM cells, CAM cells, fast PLAs, diode decoders, comparators, as well as a number of characterization devices. The first lot of wafers is just finishing fabrication and we should have characterization data in a few weeks.

The last major part of our BiCMOS effort is in creating CAD tools to support BiCMOS designs. Although we have already modified the layout tools to handle the new technology, simulation poses a much more difficult problem. To try to fill the current gap in simulation tools for BiCMOS circuits we are working on Bisim, an Rsim-like simulator for Bipolar and BiCMOS circuits. Like Rsim, Bisim is a switch level simulator, but unlike Rsim, Bisim understands that signals are voltages rather than simple boolean values. This extra flexibility allows Bisim to handle a wider class of circuits than Rsim can handle. In particular, it should be able to correctly simulate a wide class of sense circuits -- circuits that often occur in BiCMOS. At this point, we have developed piece-wise linear models for both bipolar and MOS devices. The models are simple, yet preserve the important characteristics of the devices (for digital circuits). We are now working on the evaluation routines and plan to have a version of the simulator running by the end of the school year. Our goal is to use this tool to help debug the BiCMOS RAM that we are designing.

5.2 Integrated Testers

During this period, we have continued to work on building a single chip tester. The chip, called Testarossa, contains a dRAM for the test vector storage, a decompressor to increase the effective vector size, and the pin electronics for 16 DUT pins. We have recently received the second run of these chips. This version was fabricated in a 1.6μ CMOS technology. These chips were fully functional, and preliminary testing indicates that we have enough fully functional parts to build a 256 pin, 25 MHz tester.

The vector RAM is a 1-T dRAM to increase the available vector storage. Besides using a 1-T cell, the design is quite conservative -- the cell capacitance is 0.1pf, and no lines in the array are booted above the supply. It also seems to be resonably robust. The dRAM works properly with a supply voltage of under 3.5V (a cell voltage of only about 2V). The dRAM contains 40k bits, which provides 2k raw vectors per pin. The compression algorithm we use usually compacts the vector stream by a factor of between 3-5, which means the chip can effectively store between 6-10k vectors per pin.

The decompressed test vectors are sent to the pin electronics section of the chip. This is a slightly updated version of the circuitry that we have described earlier [Gasbarro 88, Gasbarro 89]. The pin electronics allow the user to select between 5 output formats (NRZ RZ RO RTristate RComplement) and can adjust the edge placement of the output transitions to better than 1ns. The input sampling time is adjustable as well. The output driver can select between two output high levels and two output low levels to allow some pins to be tested for TTL compatability while others are driven to CMOS levels.

In addition to working on the chip, we have developed a set of software routine to make using the chip easier. These routines allow the user to easily set up and calibrate the pin electronics, and to down-load test vectors into the part. We are now working on building a proto-type 256 pin tester using 16 of these parts. This tester should run at 25MHz, and have a timing accuracy of about 1ns.

5.3 Programmable Gate Arrays

The Programmable Gate Array (PGA) is an exciting new idea in integrated circuits that reduces the IC manufacturing time from months to minutes and prototype cost from tens of thousands of dollars to under $100. A PGA is similar to a gate array in structure and purpose, but can be field-programmed to specify the function of its logic blocks and their interconnection. It was pioneered by the Xilinx Company of San Jose, California in 1986 [Carter 86] and new versions have recently been presented by that company and Actel Co. of Sunnyvale, CA [Hsieh 87, Hsieh 88, El-Ayat 88, El Gamal 88]. While PGAs are potentially a multi-billion dollar market and perhaps a boon to high-technology entrepreneurs, they also represent an entirely new area for scientific research. Indeed, the companies that have been rushing products to market have not had the time to carefully consider all of the tradeoffs involved in the design of PGAs [Carter 89].

Our initial work focuses on the design of the logic block, and was restricted to logic blocks using truth-table lookup for logic functions. It investigates the tradeoff between the functionality of the logic block (the number of inputs to the truth table), and the area required for the resulting programmable gate array. A set of industrial circuits were implemented as PGAs using tools that were developed for technology mapping, placement and routing. The implementations were done for a variety of logic blocks (different numbers of inputs) in several different programming technologies. The programming technology is the underlying method by which the truth table is filled and the interconnection is configured. The two commercial programming technologies are 1) static RAM and pass transistors [Hsieh 88] and 2) the anti-fuse [El Gamal 88].

While increasing the logic block functionality reduces the total number of logic blocks required for a circuit, it increases the area of the logic block itself and the interconnection requirements - the amount of area used for the programmable wires - because more interconnection is required in a smaller area. This latter factor results in the choice of a (truth table-based) logic block that has number of inputs of only 3 or 4. We were also able to show, for the industrial circuits we used, that it is always beneficial to include a D flip-flop in the logic block. Both of these results were shown to be independent of the programming technology. This work will appear in [Rose 89c].

Work is progressing on CAD tools for technology mapping and routing of PGAs. These present new and difficult problems due to the large number of functions each logic block can perform, and the restricted routing paths inherent in any practical programmable interconnect scheme. The tools will be used to investigate other classes of logic block designs and new interconnection structures so as to design efficient architectures for PGAs.

6 Staff

Faculty:

John Hennessy	Principal Investigator	jlh@vsop.stanford.edu
Mark Horowitz	Co-principal Investigator	horowitz@mojave.stanford.edu
Giovanni De Micheli	Assistant Professor	nanni@mojave.stanford.edu
David Dill	Assistant Professor	dill@amadeus.stanford.edu
Anoop Gupta	Assistant Professor	ag@amadeus.stanford.edu
Monica Lam	Assistant Professor	lam@mojave.stanford.edu
Daniel Weise	Assistant Professor	daniel@mojave.stanford.edu

Research Associates and Post-doctoral Scholars:

M. Ganapathi	Research Associate
K. Gopinath	Post-doctoral Scholar
S. Hwang	Research Associate
M. Morf	Visiting Scholar
S. Przybylski	Post-doctoral Scholar
J. Rose	Research Associate

Graduate Students:

R. Alverson	Electrical Engineering
J. Basu	Computer Science
M. Blatt	Computer Science
R. Chandra	Computer Science
Y. Cho	Electrical Engineering
H. Davis	Computer Science
J. Gasbarro	Electrical Engineering
K. Gharachorloo	Electrical Engineering
M. Hailpern	Computer Science
B. Hayes	Computer Science
S. Jurvetson	Electrical Engineering
R. Kao	Electrical Engineering
D. Ku	Electrical Engineering
J. Laudon	Electrical Engineering
S. McFarling	Electrical Engineering
M. Martonosi	Computer Science
S. Nowick	Computer Science
S. Richardson	Electrical Engineering
G. Rosseel	Electrical Engineering
E. Rothberg	Computer Science
A. Salz	Electrical Engineering
M. Santoro	Electrical Engineering
L. Sha	Electrical Engineering
R. Simoni	Electrical Engineering
J. Singh	Electrical Engineering

M. Smith	Electrical Engineering
L. Soule	Electrical Engineering
D. Stark	Electrical Engineering
S. Tjiang	Computer Science
A. Todesco	Electrical Engineering
J. Torrellas	Electrical Engineering
A. Tucker	Computer Science
J. Vera	Electrical Engineering
W. Weber	Computer Science
M. Wing	Electrical Engineering
D. Wingard	Electrical Engineering
M. Wolf	Electrical Engineering

Publications

[Davis 88] Davis, H., Hennessy, J.
Characterizing the Synchronization Behavior of Parallel Programs.
In *Sym. on Parallel Programming: Experience with Applications, Languages and Systems*. ACM , New Haven, CT, July, 1988.

[De Micheli 89] De Micheli, G., Klein, T.
Algorithms for Synchronous Logic Synthesis.
In *Intl. Sym. for Circuits and Systems*. IEEE, Portland, OR, May, 1989.
To appear.

[Gasbarro 88] Gasbarro, J., Horowitz, M.
Integrated Pin Electronics for VLSI Functional Testers.
In *Custom Integrated Circuits Conference*, pages 16.2.1-16.2.4. IEEE, Rochester, NY, May, 1988.

[Przybylski 88a] Przybylski, S., Horowitz, M., Hennessy, J.
Performance Effects in Memory Hierarchy Design.
In *15th International Symposium on Computer Architecture*, pages 290-298. IEEE, Honolulu, HI, June, 1988.

[Przybylski 88b] Przybylski, S.
Performance-Directed Memory Hierarchy Design.
PhD thesis, Stanford University, September, 1988.
Also published technical report number CSL-TR-88-366.

[Przybylski 89] Przybylski, S., Horowitz, M., Hennessy, J.
Characteristics of Performance-Optimal Multi-Level Cache Hierarchies.
In *16th Intl. Symposium on Computer Architecture*. IEEE, Jerusalem, Israel, May, 1989.
To appear.

[Rose 88a] Rose, J.S.
LocusRoute: A Parallel Global Router for Standard Cells.
In *25th Design Automation Conference*, pages 189-195. IEEE/ACM, Anaheim, CA, June, 1988.

[Rose 88b] Rose, J.S.
The Parallel Decomposition and Implementation of an Integrated Circuit Global Router.
In *Sigplan Symposium on Parallel Programming*, pages 138-145. ACM, New Haven, CT, July, 1988.

[Rose 88c] Rose, J., Klebsch, W., Wolf, J.
Temperature Measurement of Simulated Annealing Placements.
In *Intl. Conference on Computer-Aided Design*, pages 514-517. IEEE, November, 1988.

[Rose 89a] Rose, J.
Parallel Global Routing for Standard Cells.
IEEE Trans. on Computer-Aided Design of Circuits and Systems , 1989.
Submitted for publication.

[Rose 89b] Rose, J., Klebsch, W., Wolf, J.
Temperature Measurement and Equilibrium Dynamics of Simulated Annealing Placements.
IEEE Trans. on Computer-Aided Design of Circuits and Systems , 1989.
Submitted for publication.

[Rose 89c] Rose, J., Francis, R., Chow, P., Lewis, D.
The Effect of Logic Block Complexity on Area of Programmable Gate Arrays.
In *Custom Integrated Circuits Conference*. IEEE, 1989.
To appear.

[Smith 89] Smith, M., Johnson, M., Horowitz, M.
Limits on Multiple Instruction Issue.
In *ASPLOS-III*. ACM/IEEE, Boston, MA, April, 1989.
To appear.

[Soule 89a] Soule, L., Gupta, A.
Analysis of Parallelism and Deadlocks in Distributed-Time Logic Simulation.
1989.

[Soule 89b] Soule, L., Gupta, A.
Characterization of Parallelism and Deadlocks in Distributed Digital Logic Simulation.
In *26th Design Automation Conference*. IEEE/ACM, Las Vegas, NV, June, 1989.
To appear.

[Stark 89] Stark, D., Horowitz, M.
Techniques for Calculating Currents and Voltages in VLSI Power Supply Networks.
IEEE Transactions on Computer-Aided Design , 1989.
Submitted for publication.

[Tucker 89] Tucker, A., Gupta, A.
Process Control and Scheduling Issues for Multiprogrammed Shared-Memory Multiprocessors.
1989.

References

[Carter 86] Carter, W., et. al.
A User Programmable Reconfigurable Gate Array.
In *Custom Integrated Circuits Conference*, pages 233-235. IEEE, May, 1986.

[Carter 89] Carter, W.
Private Communication.
1989.

[Cole 89] Cole, Bernard.
Aspen Shows BiCMOS Can Yield Fast SRAMs.
Electronics :88, February, 1989.

[El Gamal 88] El Gamal, A., et. al.
An Architecture for Electrically Configurable Gate Arrays.
In *Custom Integrated Circuits Conference*, pages 15.4.1-15.4.4. IEEE, May, 1988.

[El-Ayat 88] El-Ayat, K., et. al.
A CMOS Electrically Configurable Gate Array.
In *Intl. Solid-State Circuits Conference*, pages 76-77. IEEE, 1988.

[Hsieh 87] Hsieh, H., et. al.
A Second Generation User Programmable Gate Array.
In *Custom Integrated Circuits Conference*, pages 515-521. IEEE, May, 1987.

[Hsieh 88] Hsieh, H., et. al.
A 9000-Gate User-Programmable Gate Array.
In *Custom Integrated Circuits Conference*, pages 15.3.1-15.3.7. IEEE, May, 1988.